

## Appendix A:

This appendix contains the complete source code to two algorithm modules that exemplify an embodiment of the current invention: a finite impulse response filter module (FIR) and a filter group module (FIG). Although a digital filter is much too simple an algorithm to encapsulate as a component, it illustrates (and hopefully motivates) the concepts presented in the specification. The FIR filter example consists of the following files:

- 1) fir.c, fir.h – FIR utility API module source and interface header
- 2) ifir.c, ifir.h – abstract FIR interface definition header and parameter defaults
- 3) fir\_ti.c, fir\_ti.h – vendor specific implementation and header
- 4) fir\_ti\_ext.c – vendor specific extensions to FIR
- 5) firtest.c, firtest1.c – simple programs using ALG to execute a FIR filter.

The filter group module, FIG, is an example that illustrates how multiple instances of an algorithm can be grouped together to share common coefficients.

The filter group example consists of the following files.

- 1) fig.c, fig.h – FIG utility API module source and interface header
- 2) ifig.h – abstract FIG interface definition header
- 3) fig\_ti.c, fig\_ti.h – vendor specific implementation and header
- 4) figtest.c – a simple program using ALG to execute a filter group.

Table A-1 summarizes a characterization of the performance of the FIR example, including memory usage requirements. A similar characterization can be compiled for the FIG example.

Instance Parameters	
filterlen	16
framelen	180

Other Parameters	
word size (bytes)	2
sample rate (samp/sec)	8000

Execution Time	Period	Cycles/Period
worst case	22500 us	2880

Interrupt Latency	0 cycles
-------------------	----------

Stack Memory	Size	Align
worst case	40	0

Instance Memory	DARAM		SARAM		External	
	Size	Align	Size	Align	Size	Align
scratch	390	0	0	0	0	0
persistent	0	0	0	0	42	0

Module Memory	Code		Data		BSS	
	Size	Align	Size	Align	Size	Align
fir_ti.o54	734	0	0	0	34	0
fir_ti_ext.o54	134	0	0	0	0	0
fir_ti_irtc.o54	58	0	0	0	6	0

TABLE A-1

## Name

**fir.h – FIR Module Interface**

## Text

```

/*
 * ===== fir.h =====
 * This header defines all types, constants, and functions used by
 * applications that use the FIR algorithm.
 *
 * Applications that use this interface enjoy type safety and
 * the ability to incorporate multiple implementations of the FIR
 * algorithm in a single application at the expense of some
 * additional indirection.
 */

#ifndef FIR_
#define FIR_

#include <alg.h>
#include <ifir.h>
#include <ialg.h>

/*
 * ===== FIR_Handle =====
 * FIR algorithm instance handle
 */
typedef struct IFIR_Obj *FIR_Handle;

/*
 * ===== FIR_Params =====
 * FIR algorithm instance creation parameters
 */
typedef struct IFIR_Params FIR_Params;

/*
 * ===== FIR_PARAMS =====
 * Default instance parameters
 */
#define FIR_PARAMS IFIR_PARAMS

/*
 * ===== FIR_apply =====
 * Apply a FIR filter to the input array and place results in the
 * output array.
 */
extern Void FIR_apply(FIR_Handle fir, Int in[], Int out[]);

```

[illegible]

A-4

Name

**ifir.h – Example Abstract FIR Filter Interface**

Text

```

/*
 * ===== ifir.h =====
 * This header defines all types, constants, and functions shared by all
 * implementations of the FIR algorithm.
 */
#ifndef IFIR_
#define IFIR_

#include <ialg.h>

/*
 * ===== IFIR_Obj =====
 * Every implementation of IFIR *must* declare this structure as
 * the first member of the implementation's object.
 */
typedef struct IFIR_Obj {
    struct IFIR_Fxns *fxns;
} IFIR_Obj;

/*
 * ===== IFIR_Handle =====
 * This type is a pointer to an implementation's instance object.
 */
typedef struct IFIR_Obj *IFIR_Handle;

/*
 * ===== IFIR_Params =====
 * This structure defines the parameters necessary to create an
 * instance of a FIR object.
 *
 * Every implementation of IFIR *must* declare this structure as
 * the first member of the implementation's parameter structure.
 */
typedef struct IFIR_Params {
    Int size;           /* sizeof the whole parameter struct */
    Int *coeffPtr;      /* pointer to coefficients */
    Int filterLen;      /* length of filter */
    Int frameLen;       /* length of input (output) buffer */
} IFIR_Params;

/*
 * ===== IFIR_PARAMS =====
 * Default instance creation parameters (defined in ifir.c)
 */
extern IFIR_Params IFIR_PARAMS;

```

.....

[illegible]

Name

**fir.c – Common FIR Module Implementation**

Text

```

/*
 * ===== fir.c =====
 * FIR Filter Module - implements all functions and defines all constant
 * structures common to all FIR filter algorithm implementations.
 */
#include <std.h>
#include <alg.h>

#include <fir.h>

/*
 * ===== FIR_apply =====
 * Apply a FIR filter to the input array and place results in the
 * output array.
 */
Void FIR_apply(FIR_Handle handle, Int in[], Int out[])
{
    /* activate instance object */
    ALG_activate((ALG_Handle)handle);

    handle->fxns->filter(handle, in, out);      /* filter data */

    /* deactivate instance object */
    ALG_deactivate((ALG_Handle)handle);
}

/*
 * ===== FIR_exit =====
 * Module finalization
 */
Void FIR_exit()
{
}

/*
 * ===== FIR_init =====
 * Module initialization
 */
Void FIR_init()
{
}

```

## fir\_ti.c – Vender Specific FIR Module Implementation

```

/*
 * ===== fir_ti_ialg.c =====
 * FIR Filter Module - TI implementation of a FIR filter algorithm
 *
 * This file contains an implementation of the IALG interface
 * required by XDAIS.
 */
#pragma CODE_SECTION(FIR_TI_activate, ".text:algActivate")
#pragma CODE_SECTION(FIR_TI_alloc, ".text:algAlloc()")
#pragma CODE_SECTION(FIR_TI_deactivate, ".text:algDeactivate")
#pragma CODE_SECTION(FIR_TI_free, ".text:algFree")
#pragma CODE_SECTION(FIR_TI_initObj, ".text:algInit")
#pragma CODE_SECTION(FIR_TI_moved, ".text:algMoved")

#include <std.h>

#include <ialg.h>
#include <ifir.h>
#include <fir_ti.h>
#include <fir_ti_priv.h>

#include <string.h>          /* memcpy() declaration */

#define HISTORY 1
#define WORKBUF 2
#define NUMBUFS 3

/*
 * ===== dot =====
 */
static Int dot(Int *a, Int *b, Int n)
{
    Int sum = 0;
    Int i;

    for (i = 0; i < n; i++) {
        sum += *a++ * *b++;
    }
    return (sum);
}

```



```
/*
 * ===== FIR_TI_activate =====
 * Copy filter history from external slow memory into working buffer.
 */
Void FIR_TI_activate(IALG_Handle handle)
{
    FIR_TI_Obj *fir = (Void *)handle;

    /* copy saved history to working buffer */
    memcpy((Void *)fir->workBuf, (Void *)fir->history,
           fir->filterLenM1 * sizeof(Int));
}
```

00000000"00000000

```

/*
 * ===== FIR_TI_alloc =====
 */
Int FIR_TI_alloc(const IALG_Params *algParams,
                 IALG_Fxns **pf, IALG_MemRec memTab[])
{
    const IFIR_Params *params = (Void *)algParams;

    if (params == NULL) {
        params = &IFIR_PARAMS; /* set default parameters */
    }

    /* Request memory for FIR object */
    memTab[0].size = sizeof(FIR_TI_Obj);
    memTab[0].alignment = 0;
    memTab[0].space = IALG_EXTERNAL;
    memTab[0].attrs = IALG_PERSIST;

    /*
     * Request memory filter's "inter-frame" state (i.e., the
     * delay history)
     *
     * Note we could have simply added the delay buffer size to the
     * end of the FIR object by combining this request with the one
     * above, thereby saving some code. We separate it here for
     * clarity.
     */
    memTab[HISTORY].size = (params->filterLen - 1) * sizeof(Int);
    memTab[HISTORY].alignment = 0;
    memTab[HISTORY].space = IALG_EXTERNAL;
    memTab[HISTORY].attrs = IALG_PERSIST;

    /*
     * Request memory for shared working buffer
     */
    memTab[WORKBUF].size =
        (params->filterLen - 1 + params->frameLen) * sizeof(Int);
    memTab[WORKBUF].alignment = 0;
    memTab[WORKBUF].space = IALG_DARAM0;
    memTab[WORKBUF].attrs = IALG_SCRATCH;

    return (NUMBUFS);
}

```

000260 000390



4-12

1. The first part of the report discusses the importance of maintaining accurate records of all transactions, including sales, purchases, and expenses. It emphasizes the need for consistency and transparency in financial reporting.

2. The second part of the report provides a detailed analysis of the company's financial performance over the past year. It includes a comparison of actual results against budgeted figures and identifies areas where the company has exceeded expectations.

3. The third part of the report outlines the company's financial goals for the upcoming year. It includes a discussion of the strategies that will be implemented to achieve these goals, such as increasing sales, reducing costs, and improving operational efficiency.

4. The fourth part of the report discusses the company's financial risks and the measures that will be taken to mitigate them. It includes a discussion of the company's credit policy, its approach to managing debt, and its plans for maintaining adequate liquidity.

5. The fifth part of the report provides a summary of the company's financial position and a conclusion. It reiterates the company's commitment to financial integrity and its confidence in its ability to achieve its long-term goals.

## **fir\_ti.h – Vender Specific FIR Module Interface**

**T-4**

1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399</
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	--------

```

/*
 * ===== FIR_TI_IRTC =====
 * TI's implementation of FIR's IRTC interface
 */
extern IRTC_Fxns FIR_TI_IRTC;

/*
 * ===== Vendor specific methods =====
 * The remainder of this file illustrates how a vendor can
 * extend an interface with custom operations.
 *
 * The operations below simply provide a type safe interface
 * for the creation, deletion, and application of TI's FIR filters.
 * However, other implementation specific operations can also
 * be added.
 */

/*
 * ===== FIR_TI_Handle =====
 */
typedef struct FIR_TI_Obj *FIR_TI_Handle;

/*
 * ===== FIR_TI_Params =====
 * We don't add any new parameters to the standard ones defined
 * by IFIR.
 */
typedef IFIR_Params FIR_TI_Params;

/*
 * ===== FIR_TI_PARAMS =====
 * Define our default parameters.
 */
#define FIR_TI_PARAMS    IFIR_PARAMS

/*
 * ===== FIR_TI_create =====
 * Create a FIR_TI instance object.
 */
extern FIR_TI_Handle FIR_TI_create(const FIR_TI_Params *params);

/*
 * ===== FIR_TI_delete =====
 * Delete a FIR_TI instance object.
 */
extern Void FIR_TI_delete(FIR_TI_Handle handle);

```

.....

1777 1778 1779 1780 1781 1782 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2025 2026 2027 2028 2029 2030 2031 2032 2033 2034 2035 2036 2037 2038 2039 2040 2041 2042 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126 2127 2128 2129 2130 2131 2132 2133 2134 2135 2136 2137 2138 2139 2140 2141 2142 2143 2144 2145 2146 2147 2148 2149 2150 2151 2152 2153 2154 2155 2156 2157 2158 2159 2160 2161 2162 2163 2164 2165 2166 2167 2168 2169 2170 2171 2172 2173 2174 2175 2176 2177 2178 2179 2180 2181 2182 2183 2184 2185 2186 2187 2188 2189 2190 2191 2192 2193 2194 2195 2196 2197 2198 2199 2200 2201 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 2213 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 2225 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 2237 2238 2239 2240 2241 2242 2243 2244 2245 2246 2247 2248 2249 2250 2251 2252 2253 2254 2255 2256 2257 2258 2259 2260 2261 2262 2263 2264 2265 2266 2267 2268 2269 2270 2271 2272 2273 2274 2275 2276 2277 2278 2279 2280 2281 2282 2283 2284 2285 2286 2287 2288 2289 2290 2291 2292 2293 2294 2295 2296 2297 2298 2299 2300 2301 2302 2303 2304 2305 2306 2307 2308 2309 2310 2311 2312 2313 2314 2315 2316 2317 2318 2319 2320 2321 2322 2323 2324 2325 2326 2327 2328 2329 2330 2331 2332 2333 2334 2335 2336 2337 2338 2339 2340 2341 2342 2343 2344 2345 2346 2347 2348 2349 2350 2351 2352 2353 2354 2355 2356 2357 2358 2359 2360 2361 2362 2363 2364 2365 2366 2367 2368 2369 2370 2371 2372 2373 2374 2375 2376 2377 2378 2379 2380 2381 2382 2383 2384 2385 2386 2387 2388 2389 2390 2391 2392 2393 2394 2395 2396 2397 2398 2399 2400 2401 2402 2403 2404 2405 2406 2407 2408 2409 2410 2411 2412 2413 2414 2415 2416 2417 2418 2419 2420 2421 2422 2423 2424 2425 2426 2427 2428 2429 2430 2431 2432 2433 2434 2435 2436 2437 2438 2439 2440 2441 2442 2443 2444 2445 2446 2447 2448 2449 2450 2451 2452 2453 2454 2455 2456 2457 2458 2459 2460 2461 2462 2463 2464 2465 2466 2467 2468 2469 2470 2471 2472 2473 2474 2475 2476 2477 2478 2479 2480 2481 2482 2483 2484 2485 2486 2487 2488 2489 2490 2491 2492 2493 2494 2495 2496 2497 2498 2499 2500 2501 2502 2503 2504 2505 2506 2507 2508 2509 2510 2511 2512 2513 2514 2515 2516 2517 2518 2519 2520 2521 2522 2523 2524 2525 2526 2527 2528 2529 2530 2531 2532 2533 2534 2535 2536 2537 2538 2539 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573 2574 2575 2576 2577 2578 2579 2580 2581 2582 2583 2584 2585 2586 2587 2588 2589 2590 2591 2592 2593 2594 2



Name

**fir\_ti\_priv.h – Private Vender Specific FIR Header**

Text

```

/*
 * ===== fir_ti_priv.h =====
 * Internal vendor specific (TI) interface header for FIR
 * algorithm. Only the implementation source files include
 * this header; this header is not shipped as part of the
 * algorithm.
 *
 * This header contains declarations that are specific to
 * this implementation and which do not need to be exposed
 * in order for an application to use the FIR algorithm.
 */
#ifndef FIR_TI_PRIV_
#define FIR_TI_PRIV_

#include <ialg.h>
#include <irtc.h>
#include <itst.h>
#include <ifir.h>
#include <log.h>

typedef struct FIR_TI_Obj {
    IALG_Obj    alg;          /* MUST be first field of XDAIS algs */
    IRTC_Mask   mask;        /* current test/diag mask setting */
    Int         *workBuf;    /* on-chip scratch history */
    Int         *coeff;      /* on-chip persistant coeff */
    Int         *history;    /* off chip persistant history */
    Int         filterLenM1; /* length of coefficient array - 1 */
    Int         frameLen;    /* length of input (output) buffer */
} FIR_TI_Obj;

extern LOG_Obj *FIR_TI_rtcOut; /* our output trace log */

```

```
/*
 * ===== FIR_TI_trace =====
 * Our equivalent of "printf"
 */
#define FIR_TI_trace(f, a1, a2) \
    if (FIR_TI_rtcOut != NULL) { \
        LOG_printf(FIR_TI_rtcOut, (f), (a1), (a2)); \
    }

extern Void FIR_TI_activate(IALG_Handle handle);

extern Void FIR_TI_deactivate(IALG_Handle handle);

extern Int FIR_TI_alloc(const IALG_Params *algParams, IALG_Fxns **pf,
                        IALG_MemRec memTab[]);

extern Int FIR_TI_free(IALG_Handle handle, IALG_MemRec memTab[]);

extern Int FIR_TI_initObj(IALG_Handle handle,
                          const IALG_MemRec memTab[], IALG_Handle parent,
                          const IALG_Params *algParams);

extern Void FIR_TI_moved(IALG_Handle handle,
                         const IALG_MemRec memTab[], IALG_Handle parent,
                         const IALG_Params *algParams);

extern Void FIR_TI_filter(IFIR_Handle handle, Int in[], Int out[]);

extern IRTC_Mask FIR_TI_rtcGet(IRTC_Handle handle);

extern Void FIR_TI_rtcBind(LOG_Obj *log);

extern Void FIR_TI_rtcSet(IRTC_Handle handle, IRTC_Mask mask);

#endif /* FIR_TI_PRIV_ */
```

Name

**fir\_ti\_ext.c – Vender specific FIR Extensions**

Text

```

/*
 * ===== fir_ti_ext.c =====
 */
#pragma CODE_SECTION(FIR_TI_create, ".text:create")
#pragma CODE_SECTION(FIR_TI_delete, ".text:delete")
#pragma CODE_SECTION(FIR_TI_init, ".text:init")
#pragma CODE_SECTION(FIR_TI_exit, ".text:exit")

#include <std.h>
#include <alg.h>
#include <ialg.h>
#include <fir.h>
#include <ifir.h>

#include <fir_ti.h>
#include <fir_ti_priv.h>

/*
 * ===== FIR_TI_create =====
 */
FIR_TI_Handle FIR_TI_create(const FIR_Params *params)
{
    return ((Void *)ALG_create(&FIR_TI_IALG, NULL, (IALG_Params *)params));
}

/*
 * ===== FIR_TI_delete =====
 */
Void FIR_TI_delete(FIR_TI_Handle handle)
{
    ALG_delete((ALG_Handle)handle);
}

/*
 * ===== FIR_TI_exit =====
 */
Void FIR_TI_exit(Void)
{
    ALG_exit();
}

/*
 * ===== FIR_TI_init =====
 */
Void FIR_TI_init(Void)
{
    ALG_init();
}

```

**THE** **NEW** **YORK** **PUBLIC** **LIBRARY** **ASTOR LENOX TILDEN FOUNDATION**

Name

**fir\_ti\_irtc.c – Vendor Specific Implementation of IRTC Interface**

Text

```

/*
 * ===== fir_ti_irtc.c =====
 * Filter Module IRTC implementation - TI's implementation of the
 * IRTC interface for the FIR filter algorithm
 */
#include <std.h>

#include <irtc.h>
#include <fir_ti.h>
#include <fir_ti_priv.h>
#include <log.h>

/*
 * ===== FIR_TI_rtcOut =====
 * This module's output trace log.
 */
LOG_Obj *FIR_TI_rtcOut = NULL;

/*
 * ===== FIR_TI_rtcBind =====
 */
Void FIR_TI_rtcBind(LOG_Obj *log)
{
    FIR_TI_rtcOut = log;

    FIR_TI_trace("FIR_TI_rtcBind(0x%lx)\n", log, NULL);
}

/*
 * ===== FIR_TI_rtcGet =====
 */
IRTC_Mask FIR_TI_rtcGet(IRTC_Handle handle)
{
    FIR_TI_Obj *fir = (Void *)handle;

    FIR_TI_trace("FIR_TI_rtcGet(0x%lx) = 0x%x\n", handle, fir->mask);

    return (fir->mask);
}

```

[illegible]

Name

**fir\_ti\_ifirvt.c – Vendor Specific IFIR Function Table**

Text

```

/*
 * ===== fir_ti_ifirvt.c =====
 * This file contains the function table definitions for all
 * interfaces implemented by the FIR_TI module that derive
 * from IALG
 *
 * We place these tables in a separate file for two reasons:
 * 1. We want to allow one to one to replace these tables
 *    with different definitions. For example, one may
 *    want to build a system where the FIR is activated
 *    once and never deactivated, moved, or freed.
 *
 * 2. Eventually there will be a separate "system build"
 *    tool that builds these tables automatically
 *    and if it determines that only one implementation
 *    of an API exists, "short circuits" the vtable by
 *    linking calls directly to the algorithm's functions.
 */
#include <std.h>

#include <ialg.h>
#include <ifir.h>

#include <fir_ti.h>
#include <fir_ti_priv.h>

#define IALGFXNS \
    &FIR_TI_IALG,      /* module ID */ \
    FIR_TI_activate,   /* activate */ \
    FIR_TI_alloc,      /* alloc */ \
    NULL,              /* control (NULL => no control ops) */ \
    FIR_TI_deactivate, /* deactivate */ \
    FIR_TI_free,       /* free */ \
    FIR_TI_initObj,    /* init */ \
    FIR_TI_moved,      /* moved */ \
    NULL               /* numAlloc() (NULL => IALG_MAXMEMRECS) */ \

/*
 * ===== FIR_TI_IFIR =====
 * This structure defines TI's implementation of the IFIR interface
 * for the FIR_TI module.
 */
IFIR_Fxns FIR_TI_IFIR = {          /* module_vendor_interface */
    IALGFXNS,
    FIR_TI_filter /* filter */
};

```

4-24



Name

**fir\_ti\_irtcv.t.c – Vendor Specific IRTC Function Table**

Text

```

/*
 * ===== fir_ti_irtcv.t.c =====
 * This file contains the function table definitions for the
 * IRTC interface implemented by the FIR_TI module.
 *
 * We place these tables in a separate file for two reasons:
 * 1. We want allow one to one to replace these tables
 *    with different definitions. For example, one may
 *    want to build a system where the FIR is activated
 *    once and never deactivated, moved, or freed.
 *
 * 2. Eventually there will be a separate "system build"
 *    tool that builds these tables automatically
 *    and if it determines that only one implementation
 *    of an API exists, "short circuits" the vtable by
 *    linking calls directly to the algorithm's functions.
 */
#include <std.h>

#include <irtc.h>

#include <fir_ti.h>
#include <fir_ti_priv.h>

/*
 * ===== FIR_TI_IRTC =====
 * This structure defines TI's implementation of the IRTC interface
 * for the FIR_TI module.
 */
IRTC_Fxns FIR_TI_IRTC = {
    &FIR_TI_IALG,          /* module ID */
    FIR_TI_rtcBind,        /* rtcBind */
    FIR_TI_rtcGet,          /* rtcGet */
    FIR_TI_rtcSet           /* rtcSet */
};

```

**Name**

**firtest.c – example client of FIR utility library**

**Text**

```
/*
 * ===== firtest.c =====
 * This example shows how to use the type safe FIR "utility"
 * library directly by an application.
 */
#include <std.h>
#include <fir.h>
#include <log.h>

#include <fir_ti.h>

#include <stdio.h>

extern LOG_Obj trace;

Int coeff[] = {1, 2, 3, 4, 4, 3, 2, 1};
Int input[] = {1, 0, 0, 0, 0, 0, 0};

#define FRAMELEN    (sizeof (input) / sizeof (Int))
#define FILTERLEN    (sizeof (coeff) / sizeof (Int))

Int output[FRAMELEN];

static Void display(Int a[], Int n);

/*
 * ===== main =====
 */
Int main(Int argc, String argv[])
{
    FIR_Params firParams;
    FIR_Handle fir;

    FIR_init();

    firParams = FIR_PARAMS;
    firParams.filterLen = FILTERLEN;
    firParams.frameLen = FRAMELEN;
    firParams.coeffPtr = coeff;
    if ((fir = FIR_create(&FIR_TI_IFIR, &firParams)) != NULL) {
        FIR_apply(fir, input, output);    /* filter some data */
        display(output, FRAMELEN);        /* display the result */
        FIR_delete(fir);                  /* delete the filter */
    }
    FIR_exit();

    return (0);
}
```



Name

**firtest1.c – example client of ALG, RTC, and FIR**

Text

```
/*
 * ===== firtest1.c =====
 * This example shows how the trace interface (if implemented)
 * can be used by an application. It also shows how to create
 * an algorithm instance object using the ALG interface.
 *
 * The ALG interface allows one to create code that can create
 * an instance of *any* XDAIS algorithm at the cost of a loss of
 * type safety.
 */
#include <std.h>
#include <fir.h>
#include <alg.h>
#include <log.h>
#include <ialg.h>
#include <rtc.h>

#include <fir_ti.h>

extern LOG_Obj trace;

Int coeff[] = {1, 2, 3, 4, 4, 3, 2, 1};
Int input[] = {1, 0, 0, 0, 0, 0, 0, 0};

#define FRAMELEN    (sizeof (input) / sizeof (Int))
#define FILTERLEN    (sizeof (coeff) / sizeof (Int))

Int output[FRAMELEN];

static Void display(Int a[], Int n);

/*
 * ===== main =====
 */
Int main(Int argc, String argv[])
{
    FIR_Params firParams;
    ALG_Handle alg;
    RTC_Desc rtc;

    ALG_init();
    FIR_init();
    RTC_init();

    /* bind output log to FIR_TI module */
    RTC_bind(&FIR_TI_IRTC, &trace);

    /* create an instance of a FIR algorithm */
    firParams = FIR_PARAMS;
```

```

firParams.filterLen = FILTERLEN;
firParams.frameLen = FRAMELEN;
firParams.coeffPtr = coeff;
alg = ALG_create((IALG_Fxns *)&FIR_TI_IFIR, NULL,
                (IALG_Params *)&firParams);

/* if the instance creation succeeded, create a trace descriptor */
if (alg != NULL && RTC_create(&rtc, alg, &FIR_TI_IRTC) != NULL) {

    RTC_set(&rtc, RTC_ENTER);                /* enable trace */
    FIR_apply((FIR_Handle)alg, input, output); /* filter data */
    display(output, FRAMELEN);                /* display result */

    RTC_delete(&rtc);                         /* delete rtc descriptor */
    ALG_delete(alg);                         /* delete alg instance */
}

RTC_exit();
FIR_exit();
ALG_exit();
return (0);
}

/*
 * ===== display =====
 */
static Void display(Int a[], Int n)
{
    Int i;

    for (i = 0; i < n; i++) {
        LOG_printf(&trace, "%d ", a[i]);
    }

    LOG_printf(&trace, "\n");
}

```

Name

**fig.h – Filter Group Module Interface**

Text

```

/*
 * ===== fig.h =====
 * Filter Group Module Header - This module implements a FIR
 * filter group object. A filter group object simply
 * maintains global state (common coefficients and working
 * buffer) multiple FIR objects. Thus, this module does not
 * have a "process" method, it only implements "activate"
 * "deactivate", and "getStatus".
 */
#ifndef FIG_
#define FIG_

#include <ifig.h>

typedef struct IFIG_Obj *FIG_Handle;

/*
 * ===== FIG_Params =====
 * Filter group instance creation parameters
 */
typedef struct IFIG_Params FIG_Params;

extern const FIG_Params FIG_PARAMS; /* default instance parameters */

/*
 * ===== FIG_Status =====
 * Status structure for getting FIG instance attributes
 */
typedef struct IFIG_Status FIG_Status;

/*
 * ===== FIG_activate =====
 */
extern Void FIG_activate(FIG_Handle handle);

/*
 * ===== FIG_create =====
 */
extern FIG_Handle FIG_create(IFIG_Fxns *fxns, IFIG_Params *prms);

/*
 * ===== FIG_deactivate =====
 */
extern Void FIG_deactivate(FIG_Handle handle);

/*
 * ===== FIG_delete =====
 */
extern Void FIG_delete(FIG_Handle fir);

```



Name

ifig.h – Example Abstract FIR Filter Group Interface

Text

```
/*
 * ===== ifig.h =====
 * Filter Group Module Header - This module implements a FIR filter
 * group object. A filter group object simply maintains global state
 * (common coefficients and working buffer) multiple FIR objects.
 * Thus, this module does not have a "process" method, it only
 * implements "activate" and "deactivate".
 */
#ifndef IFIG_
#define IFIG_

#include <ialg.h>

/*
 * ===== IFIG_Params =====
 * Filter group instance creation parameters
 */
typedef struct IFIG_Params {
    Int size;          /* sizeof this structure */
    Int *coeffPtr;     /* pointer to coefficient array */
    Int filterLen;     /* length of coefficient array (words) */
} IFIG_Params;

extern const IFIG_Params IFIG_PARAMS; /* default instance parameters */

/*
 * ===== IFIG_Obj =====
 */
typedef struct IFIG_Obj {
    struct IFIG_Fxns *fxns;
} IFIG_Obj;

/*
 * ===== IFIG_Handle =====
 */
typedef struct IFIG_Obj *IFIG_Handle;

/*
 * ===== IFIG_Status =====
 * Status structure for getting FIG instance attributes
 */
typedef struct IFIG_Status {
    Int *coeffPtr;     /* pointer to coefficient array */
} IFIG_Status;
```



Field	1978
Hill	1977
Hill	1976
Hill	1975
Hill	1974
Hill	1973
Hill	1972
Hill	1971
Hill	1970
Hill	1969
Hill	1968
Hill	1967
Hill	1966
Hill	1965
Hill	1964
Hill	1963
Hill	1962
Hill	1961
Hill	1960
Hill	1959
Hill	1958
Hill	1957
Hill	1956
Hill	1955
Hill	1954
Hill	1953
Hill	1952
Hill	1951
Hill	1950
Hill	1949
Hill	1948
Hill	1947
Hill	1946
Hill	1945
Hill	1944
Hill	1943
Hill	1942
Hill	1941
Hill	1940
Hill	1939
Hill	1938
Hill	1937
Hill	1936
Hill	1935
Hill	1934
Hill	1933
Hill	1932
Hill	1931
Hill	1930
Hill	1929
Hill	1928
Hill	1927
Hill	1926
Hill	1925
Hill	1924
Hill	1923
Hill	1922
Hill	1921
Hill	1920
Hill	1919
Hill	1918
Hill	1917
Hill	1916
Hill	1915
Hill	1914
Hill	1913
Hill	1912
Hill	1911
Hill	1910
Hill	1909
Hill	1908
Hill	1907
Hill	1906
Hill	1905
Hill	1904
Hill	1903
Hill	1902
Hill	1901
Hill	1900
Hill	1899
Hill	1898
Hill	1897
Hill	1896
Hill	1895
Hill	1894
Hill	1893
Hill	1892
Hill	1891
Hill	1890
Hill	1889
Hill	1888
Hill	1887
Hill	1886
Hill	1885
Hill	1884
Hill	1883
Hill	1882
Hill	1881
Hill	1880
Hill	1879
Hill	1878
Hill	1877
Hill	1876
Hill	1875
Hill	1874
Hill	1873
Hill	1872
Hill	1871
Hill	1870
Hill	1869
Hill	1868
Hill	1867
Hill	1866
Hill	1865
Hill	1864
Hill	1863
Hill	1862
Hill	1861
Hill	1860
Hill	1859
Hill	1858
Hill	1857
Hill	1856
Hill	1855
Hill	1854
Hill	1853
Hill	1852
Hill	1851
Hill	1850
Hill	1849
Hill	1848
Hill	1847
Hill	1846
Hill	1845
Hill	1844
Hill	1843
Hill	1842
Hill	1841
Hill	1840
Hill	1839
Hill	1838
Hill	1837
Hill	1836
Hill	1835
Hill	1834
Hill	1833
Hill	1832
Hill	1831
Hill	1830
Hill	1829
Hill	1828
Hill	1827
Hill	1826
Hill	1825
Hill	1824
Hill	1823
Hill	1822
Hill	1821
Hill	1820
Hill	1819
Hill	1818
Hill	1817
Hill	1816
Hill	1815
Hill	1814
Hill	1813
Hill	1812
Hill	1811
Hill	1810
Hill	1809
Hill	1808
Hill	1807
Hill	1806
Hill	1805
Hill	1804
Hill	1803
Hill	1802
Hill	1801
Hill	1800
Hill	1799
Hill	1798
Hill	1797
Hill	1796
Hill	1795
Hill	1794
Hill	1793
Hill	1792
Hill	1791
Hill	1790
Hill	1789
Hill	1788
Hill	1787
Hill	1786
Hill	1785
Hill	1784
Hill	1783
Hill	1782
Hill	1781
Hill	1780
Hill	1779
Hill	1778
Hill	1777
Hill	1776
Hill	1775
Hill	

**fig.c – Common Filter Group Module Implementation**

```

/*
 * ===== fig.c =====
 * Filter Group - this module implements a filter group; a group of FIR
 * filters that share a common set of coefficients and a working buffer.
 */
#include <std.h>
#include <fig.h>

```

```

/*
 * ===== FIG_exit =====
 */
Void FIG_exit(Void)
{

```

```

/*
 * ===== FIG_init =====
 */
Void FIG_init(Void)
{
}

```

Group - 1st Class 1st Class 2nd Class 3rd Class 4th Class 5th Class 6th Class 7th Class 8th Class 9th Class 10th Class 11th Class 12th Class 13th Class 14th Class 15th Class 16th Class 17th Class 18th Class 19th Class 20th Class 21st Class 22nd Class 23rd Class 24th Class 25th Class 26th Class 27th Class 28th Class 29th Class 30th Class 31st Class 32nd Class 33rd Class 34th Class 35th Class 36th Class 37th Class 38th Class 39th Class 40th Class 41st Class 42nd Class 43rd Class 44th Class 45th Class 46th Class 47th Class 48th Class 49th Class 50th Class 51st Class 52nd Class 53rd Class 54th Class 55th Class 56th Class 57th Class 58th Class 59th Class 60th Class 61st Class 62nd Class 63rd Class 64th Class 65th Class 66th Class 67th Class 68th Class 69th Class 70th Class 71st Class 72nd Class 73rd Class 74th Class 75th Class 76th Class 77th Class 78th Class 79th Class 80th Class 81st Class 82nd Class 83rd Class 84th Class 85th Class 86th Class 87th Class 88th Class 89th Class 90th Class 91st Class 92nd Class 93rd Class 94th Class 95th Class 96th Class 97th Class 98th Class 99th Class 100th Class

## Name

**fig\_ti.c – Vendor-Specific Filter Group Implementation**

## Text

```

/*
 * ===== fig_ti.c =====
 * Filter Group - this module implements a filter group; a group of FIR
 * filters that share a common set of coefficients and a working buffer.
 */
#pragma CODE_SECTION(FIG_TI_alloc, ".text:algAlloc()")
#pragma CODE_SECTION(FIG_TI_free, ".text:algFree()")
#pragma CODE_SECTION(FIG_TI_initObj, ".text:algInit()")
#pragma CODE_SECTION(FIG_TI_moved, ".text:algMoved()")

#include <std.h>
#include <ialg.h>
#include <fig_ti.h>
#include <ifig.h>
#include <string.h>      /* memcpy() declaration */

#define COEFF 1
#define NUMBUFS 2

typedef struct FIG_TI_Obj {
    IALG_Obj    alg;          /* MUST be first field of XDAIS algs */
    Int         *coeff;       /* on-chip persistent coefficient array */
    Int         filterLen;    /* filter length (in words) */
} FIG_TI_Obj;

/*
 * ===== FIG_TI_alloc =====
 */
Int FIG_TI_alloc(const IALG_Params *algParams, IALG_Fxns **parentFxns,
                IALG_MemRec memTab[])
{
    const IFIG_Params *params = (Void *)algParams;

    if (params == NULL) {
        params = &IFIG_PARAMS; /* set default parameters */
    }

    /* Request memory for FIG object */
    memTab[0].size = sizeof (FIG_TI_Obj);
    memTab[0].alignment = 0;
    memTab[0].space = IALG_EXTERNAL;
    memTab[0].attrs = IALG_PERSIST;

```

A-36

```

/*
 * ===== FIG_TI_getStatus =====
 */
Void FIG_TI_getStatus(IFIG_Handle handle, IFIG_Status *status)
{
    FIG_TI_Obj *fig = (Void *)handle;
    status->coeffPtr = fig->coeff;
}

/*
 * ===== FIG_TI_moved =====
 */
Void FIG_TI_moved(IALG_Handle handle,
                  const IALG_MemRec memTab[], IALG_Handle parent,
                  const IALG_Params *algParams)
{
    FIG_TI_Obj *fig = (Void *)handle;

    /* initialize the FIG object's fields */
    fig->coeff = memTab[COEFF].base;
}

```

Name

**fig\_ti.h – Vendor-Specific Filter Group Interface**

Text

```
/*
 * ===== fig_ti.h =====
 * Vendor specific (TI) interface header for Filter Group algorithm
 */
#ifndef FIG_TI_
#define FIG_TI_

#include <ialg.h>
#include <ifig.h>

/*
 * ===== FIG_TI_exit =====
 * Required module finalization function
 */
extern Void FIG_TI_exit(Void);

/*
 * ===== FIG_TI_init =====
 * Required module initialization function
 */
extern Void FIG_TI_init(Void);

/*
 * ===== FIG_TI_IALG =====
 * TI's implementation of FIG's IALG interface
 */
extern IALG_Fxns FIG_TI_IALG;

/*
 * ===== FIG_TI_IFIG =====
 * TI's implementation of FIG's IFIG interface
 */
extern IFIG_Fxns FIG_TI_IFIG;
#endif /* FIG_TI_ */
```

Name

**fig\_ti\_ifigvt.h – Vendor-Specific FIG Function Table**

Text

```

/*
 * ===== fig_ti_ifigvt.c =====
 * This file contains the function table definitions for all interfaces
 * implemented by the FIG_TI module.
 */
#include <std.h>
#include <ialg.h>
#include <ifig.h>
#include <fig_ti.h>
#include <fig_ti_priv.h>

#define IALGFXNS \
    &FIG_TI_IALG, /* implementation ID */ \
    NULL, /* activate (NULL => nothing to do) */ \
    FIG_TI_alloc, /* alloc */ \
    NULL, /* control (NULL => no control operations) */ \
    NULL, /* deactivate (NULL => nothing to do) */ \
    FIG_TI_free, /* free */ \
    FIG_TI_initObj, /* init */ \
    FIG_TI_moved, /* moved */ \
    NULL /* numAlloc() (NULL => IALG_MAXMEMRECS) */

/*
 * ===== FIG_TI_IFIG =====
 */
IFIG_Fxns FIG_TI_IFIG = { /* module_vendor_interface */
    IALGFXNS, /* IALG functions */
    FIG_TI_getStatus /* IFIG getStatus */
};

/*
 * ===== FIG_TI_IALG =====
 * This structure defines TI's implementation of the IALG interface
 * for the FIG_TI module.
 */
#ifdef _TI_
asm("_FIG_TI_IALG .set _FIG_TI_IFIG");
#else

```

.....

[illegible]



Name

**fig\_ti\_priv.h – Private Vendor-Specific Filter Group Header**

Text

```

/*
 * ===== fig_ti_priv.h =====
 * Internal vendor specific (TI) interface header for FIG
 * algorithm. Only the implementation source files include
 * this header; this header is not shipped as part of the
 * algorithm.
 *
 * This header contains declarations that are specific to
 * this implementation and which do not need to be exposed
 * in order for an application to use the FIG algorithm.
 */
#ifndef FIG_TI_PRIV
#define FIG_TI_PRIV

#include <ialg.h>

typedef struct FIG_TI_Obj {
    IALG_Obj    alg;           /* MUST be first field of XDAIS algs */
    Int         *coeff;        /* on-chip persistent coefficient array */
    Int         filterLen;     /* filter length (in words) */
} FIG_TI_Obj;

extern Int FIG_TI_alloc(const IALG_Params *, IALG_Fxns **, IALG_MemRec *);
extern Int FIG_TI_free(IALG_Handle, IALG_MemRec *);
extern Void FIG_TI_getStatus(IFIG_Handle handle, IFIG_Status *status);
extern Int FIG_TI_initObj(IALG_Handle,
    const IALG_MemRec *, IALG_Handle, const IALG_Params *);
extern Void FIG_TI_moved(IALG_Handle,
    const IALG_MemRec *, IALG_Handle, const IALG_Params *);

#endif

```

Name

figtest.c – Example Client of FIG and ALG

Text

```

/*
 * ===== figtest.c =====
 * Example use of FIG, FIR and ALG modules. This test creates some
 * number of FIR filters that all share a common set of coefficients
 * and working buffer. It then applies the filter to the data and
 * displays the results.
 */
#include <std.h>
#include <fig.h>
#include <fir.h>
#include <log.h>

#include <fig_ti.h>
#include <fir_ti.h>

extern LOG_Obj trace;

#define NUMFRAMES 2 /* number of frames of data to process */

#define NUMINST 4 /* number of FIR filters to create */
#define FRAMELEN 7 /* length of in/out frames (words) */
#define FILTERLEN 8 /* length of coeff array (words) */

Int coeff[FILTERLEN] = { /* filter coefficients */
    1, 2, 3, 4, 4, 3, 2, 1
};

Int in[NUMINST][FRAMELEN] = { /* input data frames */
    {1, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 0}
};

Int out[NUMINST][FRAMELEN]; /* output data frames */

static Void display(Int a[], Int n);

/*
 * ===== main =====
 */
Int main(Int argc, String argv[])
{
    FIG_Params figParams;
    FIR_Params firParams;
    FIG_Status figStatus;
    FIG_Handle group;
    FIR_Handle inst[NUMINST];
    Bool status;

```

```

Int i, n;

FIG_init();
FIR_init();

figParams = FIG_PARAMS;
figParams.filterLen = FILTERLEN;
figParams.coeffPtr = coeff;

/* create the filter group */
if ((group = FIG_create(&FIG_TI_IFIG, &figParams)) != NULL) {

    /* get FIG pointers */
    FIG_getStatus(group, &figStatus);

    /* create multiple filter instance objects that reference group */
    firParams = FIR_PARAMS;
    firParams.frameLen = FRAMELEN;
    firParams.filterLen = FILTERLEN;
    firParams.coeffPtr = figStatus.coeffPtr;
    for (status = TRUE, i = 0; i < NUMINST; i++) {
        inst[i] = FIR_create(&FIR_TI_IFIR, &firParams);
        if (inst[i] == NULL) {
            status = FALSE;
        }
    }
    /* if object creation succeeded, apply filters to data */
    if (status) {
        /* activate group object */
        FIG_activate(group);

        /* apply all filters on all frames */
        for (n = 0; n < NUMFRAMES; n++) {
            for (i = 0; i < NUMINST; i++) {
                FIR_apply(inst[i], in[i], out[i]);
                display(out[i], FRAMELEN);
            }
        }
        /* deactivate group object */
        FIG_deactivate(group);
    }

    /* delete filter instances */
    for (i = 0; i < NUMINST; i++) {
        FIR_delete(inst[i]);
    }

    /* delete filter group object */
    FIG_delete(group);
}
FIG_exit();
FIR_exit();

```

Year	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100
1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	